Appunti di matematica applicata al calcolatore

Algebra di Boole
Routine di calcolo in assembly

ALGEBRA DI BOOLE.

Un insieme $B = \{a,b,c,...\}$ sul quale sono definite le operazioni binarie $+$ e $\cdot$ si chiama algebra di Boole se per esso valgono i seguenti assiomi:

1. Le operazioni binarie $+$ e $\cdot$ godono della proprieta' commutativa, cioe' e':

$$a + b = b + a \qquad a \cdot b = b \cdot a$$

2. L'insieme B ha due elementi particolari: un primo elemento 0 ed un ultimo elemento 1 tali che sia:

$$a + 0 = a \qquad a \cdot 1 = a$$

3. Ciascuna operazione gode della proprieta' distributiva rispetto all'altra, cioe' e':

$$a + (b \cdot c) = (a + b) \cdot (a + c)$$

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

4. Per ogni elemento $a \in B$ esiste un elemento complementare $\overline{a}$ tale che:

$$a + \overline{a} = 1 \qquad a \cdot \overline{a} = 0$$

La simmetria dei postulati rispetto alle operazioni $+$ e $\cdot$ e rispetto agli elementi 0 e 1 giustifica il:

(principio di dualita') - ogni relazione deducibile dai postulati di un'algebra di Boole rimane valida se si scambiano tra loro i simboli $+$ e $\cdot$ e gli elementi particolari 0 e 1.

Come conseguenza bastera' dimostrare solo una fra due relazioni duali.

Dimostriamo che:

per ogni $a \in B$ e' $a + a = a$ e $a \cdot a = a$.

E': $a + a = (a + a) \cdot 1 = (a + a) \cdot (a + \overline{a}) = a + (a + \overline{a}) = a + 0 = a$
Per il principio di dualita' e' vera anche la relazione duale.

Dimostriamo che:

per ogni $a \in B$ e': $a + 1 = 1$ e $a \cdot 0 = 0$.

E': $a \cdot 0 = 0 + (a \cdot 0) = (a \cdot \overline{a}) + (a \cdot 0) = a \cdot (\overline{a} + 0) = a \cdot \overline{a} = 0$
Per il principio di dualita' e' vera anche la relazione duale.

Dimostriamo che:

per ogni a,b $\in$ B e': $a + (a \cdot b) = a$ e $a \cdot (a + b) = a$.

E': $a + (a \cdot b) = (a \cdot 1) + (a \cdot b) = a \cdot (1 + b) = a \cdot 1 = a$.
Per il principio di dualita' e' vera anche la relazione duale.

Valgono le seguenti relazioni:

(proprieta' associativa): $(a + b) + c = a + (b + c)$
$(a \cdot b) \cdot c = a \cdot (b \cdot c)$

(teoremi di De Morgan): $\overline{a + b} = \bar{a} \cdot \bar{b}$

$\overline{a \cdot b} = \bar{a} + \bar{b}$

Dimostriamo che:

per ogni a,b $\in$ B e': $a + \bar{a} \cdot b = a + b$.

E': $a + \bar{a} \cdot b = a \cdot (a + b) + \bar{a} \cdot b = a \cdot a + a \cdot b + \bar{a} \cdot b =$

$= a + b \cdot (a + \bar{a}) = a + b \cdot 1 = a + b$.

Se B e' costituito da due soli elementi, B = {a,b}, e':

$a + b = 1$ $\qquad$ $a \cdot b = 0$.

Se B = {a,b,c} e': $a + b + c = 1$, $a \cdot b \cdot c = 0$.

Ogni espressione contente un numero finito di elementi di un'algebra di Boole collegati dalle operazioni + e $\cdot$ si chiama funzione di Boole.

Il numero di variabili in ciascuna funzione e' dato dal numero delle lettere distinte che in essa compaiono senza tenere conto se sono o non sono soprassegnate.
Ad esempio:

$x \cdot \bar{x}$ e' funzione di una variabile

$x \cdot \bar{y}$ e' funzione di due variabili.

Funzioni e tabelle di verita'.

Limitiamoci al caso B = {0,1}.

Le tavole della verita' sono tabelle sulle quali si riportano tutte le combinazioni binarie delle variabili che si considerano ed i risultati delle operazioni che con esse si vogliono fare.

Funzione NOT: $f = \overline{a}$

a	$\overline{a}$
0	1
1	0

Funzione AND: $f = a \cdot b$

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

Funzione OR: $f = a + b$

a	b	$a+b$
0	0	0
0	1	1
1	0	1
1	1	1

Funzione NAND: $f = \overline{a \cdot b}$

a	b	$\overline{a \cdot b}$
0	0	1
0	1	1
1	0	1
1	1	0

Funzione NOR: $f = \overline{a + b}$

a	b	$\overline{a+b}$
0	0	1
0	1	0
1	0	0
1	1	0

Le tavole della verita' sono il mezzo piu' semplice per dimostrare i teoremi dell'algebra booleana. Vediamo alcuni esempi.

(Teorema della doppia inversione) - Una variabile invertita due volte riprende il suo valore primitivo: $\bar{\bar{a}} = a$.

a	$\bar{a}$	$\bar{\bar{a}}$
0	1	0
1	0	1

(Primo teorema di De Morgan) - $\overline{a \cdot b} = \bar{a} + \bar{b}$.

La dimostrazione e' data dalla seguente tavola:

a	b	$\bar{a}$	$\bar{b}$	$a \cdot b$	$\overline{a \cdot b}$	$\bar{a} + \bar{b}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

(Secondo teorema di De Morgan) - $\overline{a + b} = \bar{a} \cdot \bar{b}$.

La dimostrazione e' data dalla seguente tavola:

a	b	$\bar{a}$	$\bar{b}$	$a+b$	$\overline{a+b}$	$\bar{a} \cdot \bar{b}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Dall'algebra di Boole all' elettronica digitale.

L'elemento base dell'attuale tecnologia elettronica digitale e' il transistor. Supponiamo di avere un transistor NPN con l'emettitore collegato a massa ed il collettore collegato a +5V attraverso un resistore. Se sulla base del transistor e' presente tensione positiva il transistor conduce e sul collettore e' presente una tensione prossima a 0V. Se al contrario sulla base e' presente una tensione di 0V, il transistor non conduce e quindi sul collettore vi sara' una tensione positiva di 5V.
Si intuisce, dunque, il possibile uso logico di questo elemento: se consideriamo livelli di tensione prossimi a 0V come 0 e livelli positivi superiori ad una certa soglia come 1, il transistor, considerata come ingresso la base e come uscita il collettore, puo' essere usato per realizzare un invertitore, cioe' un dispositivo che realizzi la funzione NOT.

Con due transistor, collegati in modo che l'emettitore dell'uno sia collegato con il collettore del secondo, si puo' realizzare la funzione NAND, e quindi usando NOT e NAND si possono realizzare facilmente le altre funzioni, grazie al teorema della doppia inversione ed ai teoremi di De Morgan.

Avendo a disposizione dispositivi che realizzino le funzioni logiche, si puo' facilmente costruire un dispositivo sommatore. Se x,y sono le cifre binarie in ingresso a tale dispositivo, e z e c (carry = riporto) sono le cifre in uscita, si puo' scrivere la seguente tavola di verita':

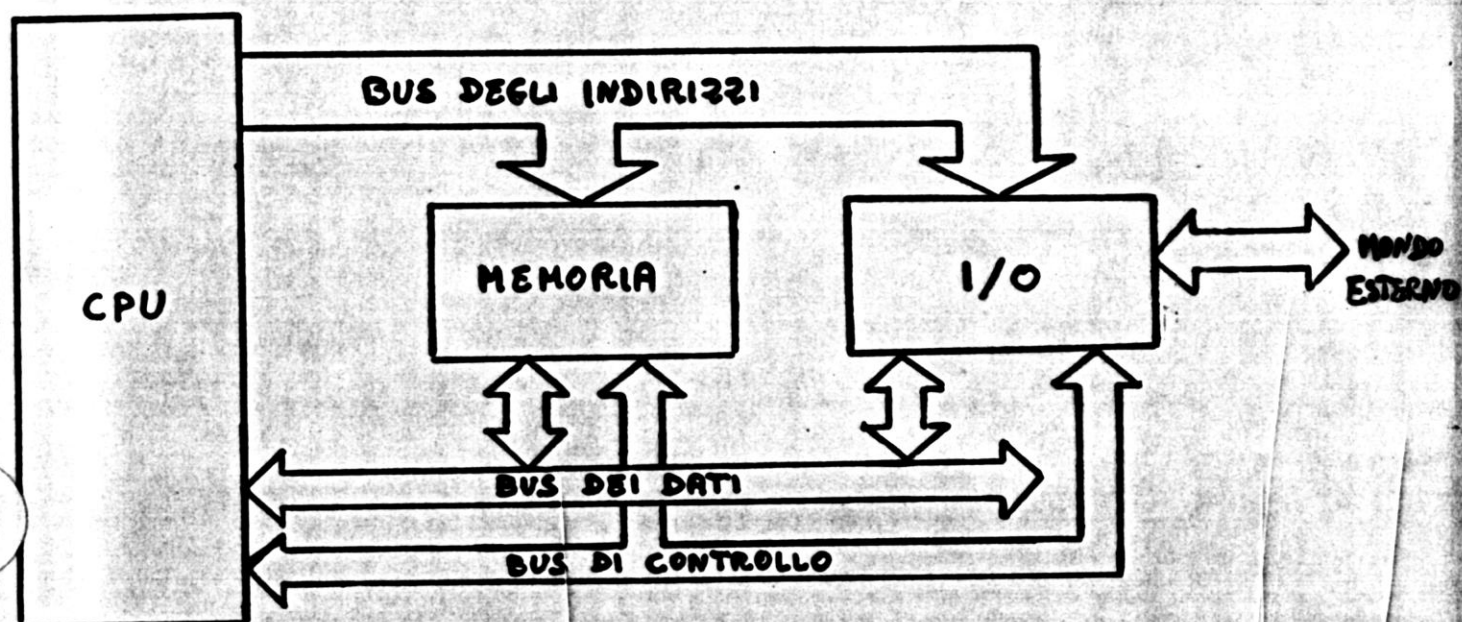
x	y	c	z
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Osserviamo che c puo' essere ottenuto realizzando un circuito che fornisca in uscita $x\cdot y$, mentre z e' dato dall'OR esclusivo, che puo' essere realizzato come $x\bar{y} + \bar{x}y$.
Applicando i teoremi di De Morgan si ha:

$$x\bar{y} + \bar{x}y = \overline{\overline{x\cdot\bar{y}} \cdot \overline{\bar{x}\cdot y}}$$

ed e' anche:

$$xy = \overline{\overline{x\cdot y}}$$

Osservando che $\overline{x\cdot x} = \bar{x}$ per realizzare il circuito che sommi due bit si possono usare 7 blocchi NAND.

CPU (CENTRAL PROCESSING UNIT):

UN SOLO CIRCUITO INTEGRATO CHE ATTUA IL TRASFERIMENTO DEI DATI, IL CONTROLLO, LE OPERAZIONI ARITMETICHE, LOGICHE E DI GESTIONE DELLE INTERRUZIONI, IN CONSEGUENZA DELL'ESECUZIONE DI ISTRUZIONI PRELEVATE DALLA MEMORIA.

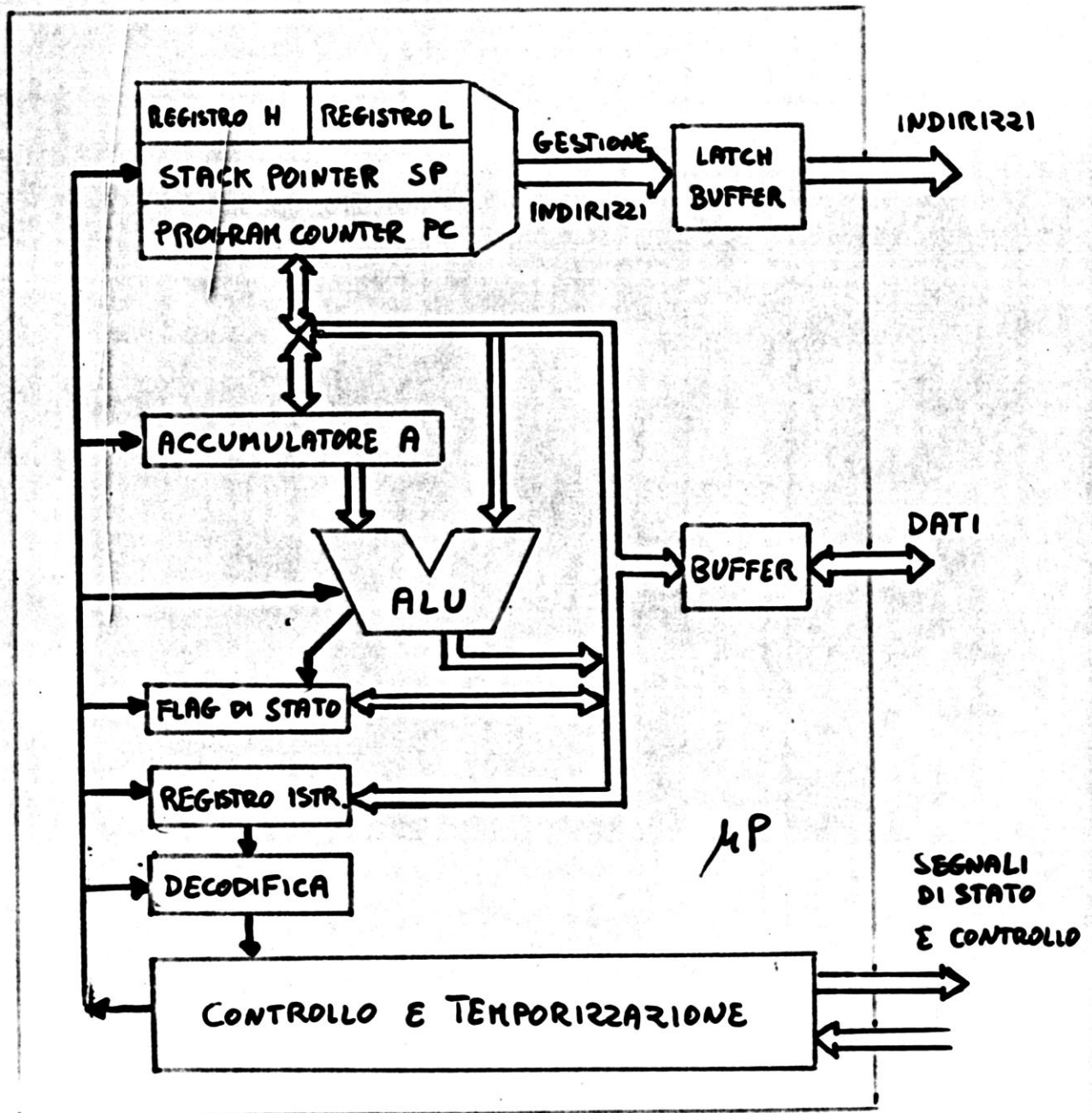
MEMORIA:

UN QUALUNQUE DISPOSITIVO CHE POSSA MEMORIZZARE BIT DI LIVELLO LOGICO 0 E 1, IN MODO CHE UN SINGOLO BIT O UN GRUPPO DI BIT (PAROLA) POSSANO VENIRE LOCALIZZATI E ESTRATTI DALLA MEMORIA STESSA.

DISPOSITIVO DI I/O:

QUALUNQUE DISPOSITIVO DIGITALE, ANCHE UN SOLO CIRCUITO INTEGRATO, CHE TRASMETTA DATI A UN CALCOLATORE, O CHE RICEVA DATI O IMPULSI DI STROBE DA UN CALCOLATORE.

STRUTTURA INTERNA DI UN μP.

LE STRUTTURE INTERNE DEI VARI μP SONO ANCHE NOTEVOLMENTE DIVERSE TRA LORO. QUI SI PRESENTA UNA STRUTTURA TIPICA SEMPLIFICATA EVIDENZIANDO I BLOCCHI INTERNI CORRISPONDENTI ALLE FUNZIONI NORMALMENTE PRESENTI IN TUTTI I μP.

80 pin configuration

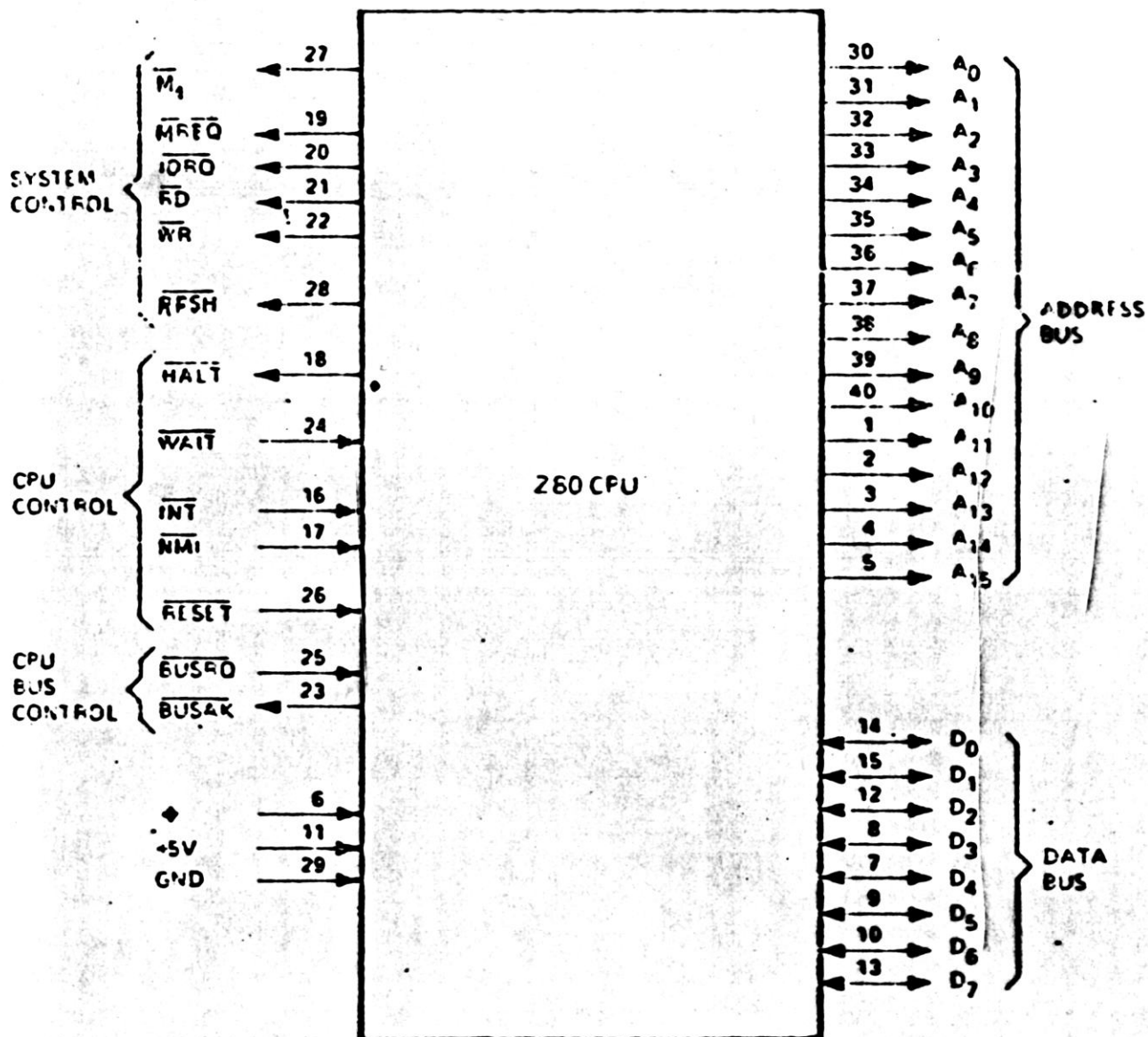



Fig 4 0-0 - CPU basic timing example

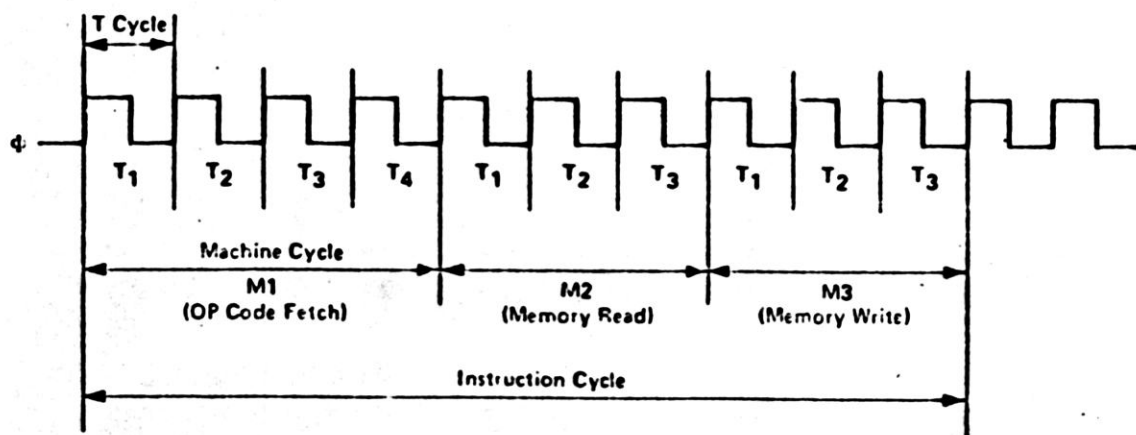
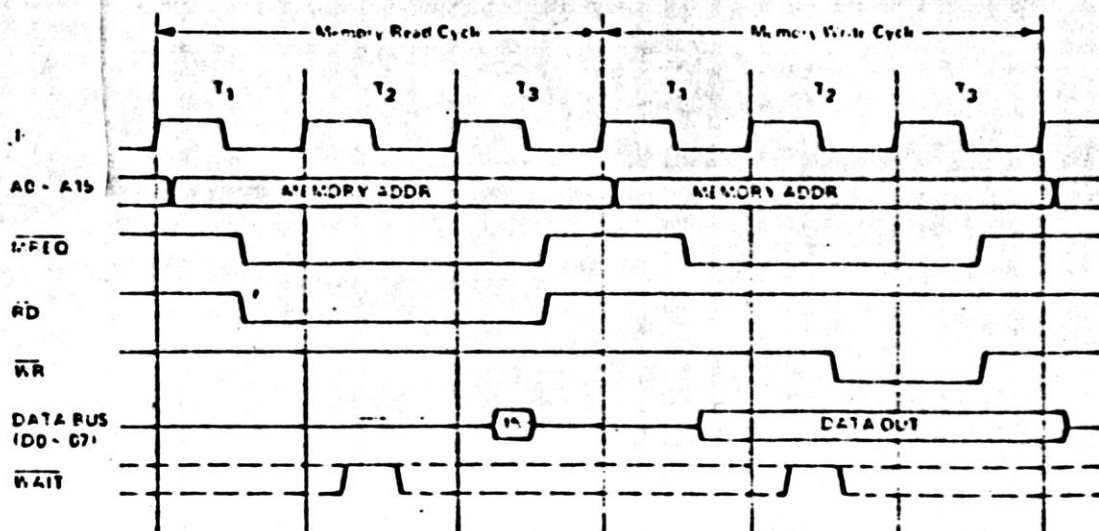
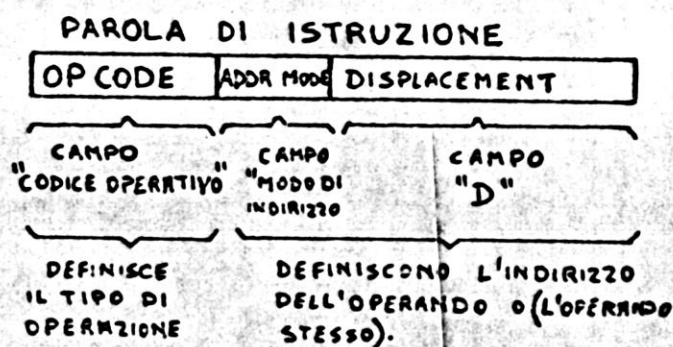



Fig. 4 0-2 - Memory read or write cycles

STRUTTURA DELLE ISTRUZIONI

ALL'INTERNO DI OGNI ISTRUZIONE E' SPECIFICATO :

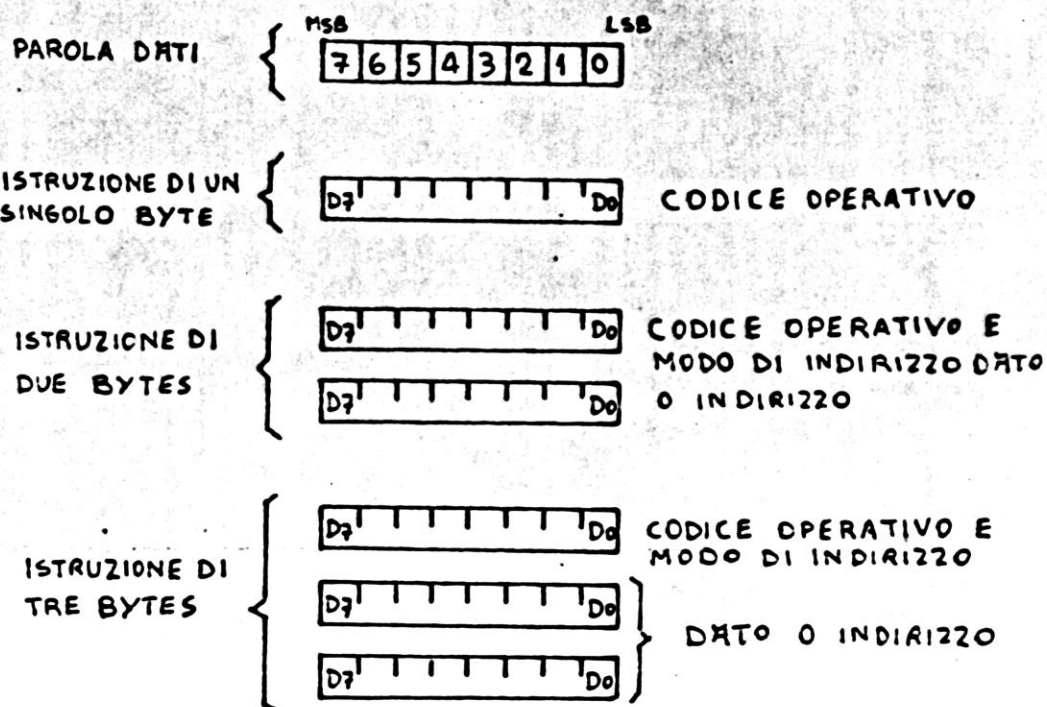
- IL TIPO DI OPERAZIONE
- IL MODO DI INDIRIZZAMENTO

LE ISTRUZIONI SONO DIVISE IN CAMPI (FIELDS)

ESEMPIO DI PAROLA-DATI E DI FORMATO ISTRUZIONI PER MICROPROCESSOR DA 8 BITS.

PAROLA DATI { MSB | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | LSB

ISTRUZIONE DI UN SINGOLO BYTE { D_7 … D_0 — CODICE OPERATIVO

ISTRUZIONE DI DUE BYTES { D_7 … D_0 — CODICE OPERATIVO E MODO DI INDIRIZZO; D_7 … D_0 — DATO O INDIRIZZO

ISTRUZIONE DI TRE BYTES { D_7 … D_0 — CODICE OPERATIVO E MODO DI INDIRIZZO; D_7 … D_0, D_7 … D_0 } DATO O INDIRIZZO

Set di istruzioni del microprocessore Z80.

(Trasferimento) - Si hanno istruzioni in grado di trasferire parole di 8 bit (byte) tra registri della CPU e tra registro e cella di memoria. Alcune istruzioni operano su coppie di bytes (16 bit).
Esempi:

LD A,7AH carica nel registro A la configurazione 01111010

LD (HL),B trasferisce B nella cella di memoria puntata da HL

(Aritmetiche-logiche) - Si hanno istruzioni che effettuano operazioni aritmetiche (somma e sottrazione) e logiche (AND, OR, NOT, XOR).
Esempi:

ADD A,C carica in A il risultato della somma A + C

OR (HL) carica in A la somma logica di A con il contenuto della cella di memoria puntata da HL.

(Rotazione e traslazione) - Si hanno istruzioni che usano i registri come registri a scorrimento (shift register). Sono utili ad esempio per moltiplicare e dividere numeri binari (una traslazione corrisponde a una moltiplicazione o divisione per due).
Esempio:

SLA D il registro D viene fatto scorrere di una posizione verso sinistra. Il bit piu' significativo viene portato nel flag CY e quello meno significativo viene azzerato.

(Bit) - Si hanno istruzioni di set, reset e test di singoli bit di un operando.
Esempio:

BIT 5,H carica nel flag Z il complemento del bit di potenza 5 di H.

(Salto) - Si hanno istruzioni di salto incondizionato o condizionato al valore di un flag.
Esempi:

JP 27A2H carica la configurazione 0010011110100010 nel PC (program counter), quindi "salta" a quella locazione di memoria.

JR C,42 se il flag CY contiene 1, somma la configurazione 01000010 al valore attuale del PC, altrimenti prosegue senza modificare nulla.

(Chiamata e ritorno [per subroutines]) - Le istruzioni di chiamata salvano nello stack il contenuto attuale del PC (cioe' l'indirizzo di rientro), aggiornano lo SP (stack pointer) e forzano nel PC il punto di inizio della subroutine. Le istruzioni di ritorno prelevano dallo stack l'indirizzo di rientro, lo forzano nel PC e aggiornano lo SP.
Esempi:

CALL 21AAH esegue la subroutine che inizia nella posizione 21AA della memoria.

RET NZ termina la subroutine e ritorna al programma chiamante se il flag Z contiene 0, altrimenti prosegue.

(Input-output) - Si hanno istruzioni che trasferiscono dati tra i registri della CPU e le porte di ingresso-uscita (identificate mediante 8 linee di indirizzamento).
Esempio:

IN A,(4EH) carica in A il dato prelevato dalla porta di ingresso di indirizzo 01001110.

(Operazioni su blocchi) - Lo Z80 possiede anche istruzioni in grado di effettuare confronti o trasferimenti fra gruppi di celle di memoria usando un registro della CPU come contatore.
Esempio:

LDIR il contenuto della cella puntata da HL viene trasferito nella cella puntata da DE; HL e DE vengono incrementati, mentre BC viene decrementato. Se BC=0, l'istruzione ha termine, altrimenti si ricomincia.

(Istruzioni di controllo) - Sono istruzioni che modificano il modo di funzionamento della CPU o manipolano le sue informazioni di stato interno.
Esempi:

NOP nessuna operazione viene eseguita.
HALT forza la CPU ad eseguire un ciclo infinito di NOP.
EI abilita gli interrupt.
DI disabilita gli interrupt.

I registri dello Z80.

```
 set principale         set alternativo

|-------|-------|      |-------|-------|
|   A   |   F   |      |   A'  |   F'  |
|-------|-------|      |-------|-------|
|   B   |   C   |      |   B'  |   C'  |
|-------|-------|      |-------|-------|
|   D   |   E   |      |   D'  |   E'  |
|-------|-------|      |-------|-------|
|   H   |   L   |      |   H'  |   L'  |
|-------|-------|      |-------|-------|

|-------|-------|
|   I   |   R   |
|-------|-------|
|      IX       |
|-------|-------|
|      IY       |
|-------|-------|
|      SP       |
|-------|-------|
|      PC       |
|-------|-------|
```

Il contatore di programma (PC) contiene l'indirizzo dell'operazione successiva che deve essere eseguita. Un semplice incrementatore connesso al contatore di programma fornisce il meccanismo automatico per prelevare le istruzioni in sequenza dalla memoria.

Lo Z80 possiede un set duplicato di registri di uso generale. Due istruzioni exchange selezionano e deselezionano tutti i registri alternativi. Una volta effettuata la scelta, tutte le operazioni successive sui registri sono effettuate sull'insieme attivo finche' il prossimo scambio non scelga l'insieme non attivo.

L'accumulatore (A) ha rispetto agli altri registri ad 8 bit una sua particolare importanza. Innanzitutto tutte le istruzioni aritmetiche e booleane ad 8 bit richiedono un operando dall'accumulatore e riportano il risultato nell'accumulatore. Inoltre i trasferimenti di dati piu' corti e veloci tra la CPU e i dispositivi di I/O sono realizzati mediante l'accumulatore. Infine esiste un numero maggiore di istruzioni di riferimento alla memoria che spostano dati tra accumulatore e memoria piuttosto che tra un altro registro e memoria.

Ai registri B,C,D,E,H,L si puo' accedere con uguale facilita' per qualunque di essi. I dati in essi contenuti possono essere spostati verso qualunque altro registro o usati come secondo operando in istruzioni a due operandi. Esistono pero' tra essi alcune importanti differenze. I registri H ed L sono il principale puntatore di dato per lo Z80. Di regola si usano questi due registri per conservare l'indirizzo di memoria a 16

bit del dato da indirizzare. L'uso della coppia HL come puntatore di dato spesso richiede meno byte di codice oggetto e meno cicli di istruzione per compiere operazioni mediante essi. Esiste un numero limitato di istruzioni che trattano i registri B e C, D e E come puntatore di dato a 16 bit. Queste istruzioni pero' spostano dati solo tra memoria e l'accumulatore.
Alcune operazioni aritmetiche accedono a BC, DE, HL come unita' di dato a 16 bit.

I registri IX, IY sono registri indice. Forniscono una limitata capacita' di indicizzamento.

Il puntatore di stack (SP) a 16 bit permette di effettuare una operazione di stack dovunque nella memoria indirizzabile. Lo stack e' una struttura LIFO (last-in, first-out). E' una struttura cronologica: il primo elemento introdotto nello stack e' sempre nella parte piu' bassa dello stack, mentre l'elemento depositato piu' recentemente nello stack e' sulla sommita' dello stack. Lo stack e' richiesto per rendere disponibili tre possibilita' di programmazione: sottoprogrammi, interrupt e immagazzinamento temporaneo di dati contenuti nei registri.

Il ruolo del registro dei flag (F) e' quello di memorizzare il verificarsi di condizioni eccezionali dentro il microprocessore.

```
   7   6   5   4   3   2   1   0
 |---|---|---|---|---|---|---|---|
 | S | Z |   | H |   |P/V| N | C |
 |---|---|---|---|---|---|---|---|
```

Il flag carry (C) riporta il bit piu' significativo di una qualunque operazione aritmetica. E' azzerato da istruzioni booleane.

Il flag subtract (N) e' stato progettato per uso interno durante operazioni di regolazione decimale. E' 1 per tutte le istruzioni subtract, 0 per tutte le istruzioni add.

Il flag parity/overflow (P/V) e' un flag di overflow per le operazioni aritmetiche ed e' flag di parita' (1=parita' pari, 0=parita' dispari) per le operazioni di input, di rotazione e booleane.

Il flag half-carry (H) mantiene ogni riporto dal bit 3 al bit 4 risultante dall'esecuzione di un'istruzione aritmetica. Lo scopo di questo flag e' di semplificare le operazioni BCD.

Il flag zero (Z) e' posizionato a 1 quando una operazione aritmetica o booleana genera un risultato zero; se il risultato e' diverso da zero il flag e' posizionato a 0.

Il flag sign (S) acquisisce il valore del bit piu' significativo del risultato conseguente l'esecuzione di un'operazione aritmetica o booleana.

La maggior parte delle istruzioni eseguite dal microprocessore modifichera' alcuni o tutti i flag. I flag mantengono il loro valore corrente finche' non sia eseguita una istruzione che li modifichi.

Routines aritmetiche in linguaggio assembly.

Somma ad 8 bit.

Sommare il contenuto di due celle di memoria e porre il risultato in una terza cella. Le tre celle hanno indirizzi consecutivi.

```
ADDEN1 EQU 200H
       ORG 100H
START:
       LD HL,(ADDEN1)      ;si trasferiscono in H e L i due addendi
       LD A,L
       ADD A,H
                           ;si e' eseguita la somma
       LD (ADDEN1+2),A     ;si memorizza il risultato
       HALT
```

Somma a 16 bit di dati a 16 bit.

Calcolare la somma a 16 bit di una serie di dati a 16 bit. Nelle locazioni RISULT e RISULT+1 verra' memorizzato il risultato con gli otto bit meno significativi nella prima delle due locazioni. La lunghezza della serie e' contenuta nella locazione RISULT+2 e la serie e' memorizzata a partire da RISULT+3. Ciascun dato a 16 bit occupa due celle di memoria, con il byte piu' significativo nell'indirizzo piu' elevato.

```
RISULT   EQU 200H
         ORG 100H
START:
         LD HL,RISULT+2
         LD B,(HL)        ;inizializzazione del contatore di dati
         LD D,0
         LD E,D           ;azzeramento dei registri che ospiteranno
                          ;le somme parziali
CICLO:
         INC HL           ;puntatore sul byte meno significativo
                          ;del dato
         LD A,E
         ADD A,(HL)
         LD E,A           ;si e' eseguita la somma del byte meno
                          ;significativo
         INC HL           ;puntatore sul byte piu' significativo
         LD A,D
         ADC A,(HL)
         LD D,A           ;si e' eseguita la somma del byte piu'
                          ;significativo
         DJNZ CICLO       ;si continua fino a che vi sono dati
         LD (RISULT),DE   ;memorizzazione del risultato
         HALT
```

Somma in multiprecisione.

Sommare due numeri binari multi-byte. La lunghezza dei numeri binari in byte e' contenuta nella locazione 200H, mentre i numeri iniziano con il byte meno significativo dalle 201H e 211H rispettivamente, e la somma sostituisce il numero che inizia nella cella 201H.

```
        ORG 100H
START:
        LD HL,200H
        LD B,(HL)    ;contatore di byte = lunghezza
        INC HL       ;puntatore all'inizio del primo numero
        LD DE,211H   ;puntatore all'inizio del secondo numero
        AND A        ;si inizializza il carry a zero
CICLO:
        LD A,(DE)
        ADC A,(HL)
        LD (HL),A    ;effettuata la somma si memorizza nelle
                     ;locazioni di memoria occupate dal primo
                     ;numero
        INC DE
        INC HL
        DJNZ CICLO
        HALT
```

Moltiplicazione binaria ad 8 bit.

Moltiplicare il numero ad 8 bit privo di segno contenuto nella locazione di memoria 200H con il numero ad 8 bit privo di segno contenuto nella locazione 201H. Porre gli otto bit meno significativi del risultato nella cella 202H e gli otto bit piu' significativi nella cella 203H.

```
        ORG 100H
START:
        LD DE,(200H)
        LD A,D
        LD D,0
        LD H,D
        LD L,D          ;si ha in A il moltiplicatore, in DE il
                        ;moltiplicando esteso a 16 bit, in HL
                        ;prodotto = 0
        LD B,8          ;contatore = lunghezza in bit del
                        ;moltiplicatore
LOOP:
        ADD HL,HL       ;scorrimento del prodotto a sinistra di
                        ;un bit
        RLA             ;scorrimento del moltiplicatore a
                        ;sinistra di un bit
        JR NC,EXLOOP
        ADD HL,DE       ;si somma il moltiplicando al prodotto
                        ;solo se si ha carry dallo scorrimento
                        ;del moltiplicatore a sinistra
```

```
EXLOOP:
         DJNZ LOOP
         LD (202H),HL
         HALT
```

L'algoritmo usato e' il seguente:

```
BEGIN
     moltiplicando = (200H)
     moltiplicatore = (201H)
     prodotto = 0
     contatore = 8
     DO WHILE contatore > 0
        prodotto = 2 * prodotto
        moltiplicatore = 2 * moltiplicatore
        IF carry proviene da moltiplicatore
           THEN
                prodotto = prodotto + moltiplicando
        ENDIF
        contatore = contatore - 1
     REPEAT
     (202H e 203H) = prodotto
END
```

__Divisione binaria ad 8 bit.__

Dividere il numero privo di segno a 16 bit contenuto nelle celle di memoria 200H e 201H (gli otto bit piu' significativi in 201H) per il numero privo di segno ad 8 bit contenuto nella cella di memoria 202H.
I numeri sono normalizzati in modo che: i bit piu' significativi sia del dividendo che del divisore siano zero; il numero presente nella cella 202H sia maggiore del numero presente nella cella 201H, vale a dire che il quoziente sia un numero ad 8 bit.
Memorizzare il quoziente nella cella 203H e il resto della divisione in 204H.

```
        ORG 100H
START:
        LD HL,(200H)
        LD A,(202H)
        LD C,A
        LD B,8
LOOP:
        ADD HL,HL
        LD A,H
        SUB C
        JR C,EXLOOP
        LD H,A
        INC L
EXLOOP:
        DJNZ LOOP
        LD (203H),HL
        HALT
```

L'algoritmo usato e' il seguente:

```
BEGIN
     dividendo = (200H e 201H)
     divisore = (202H)
     contatore = 8
     quoziente = 0
     dividendo = 2 * dividendo
     quoziente = 2 * quoziente
     DO WHILE contatore > 0
        IF divisore <= msb(dividendo)
           THEN
               msb(dividendo) = msb(dividendo) - divisore
               quoziente = quoziente + 1
        ENDIF
        contatore = contatore - 1
     REPEAT
     (203H) = quoziente
     (204H) = msb(dividendo)
END
```

Quadrato di un numero.

Calcolare il quadrato del numero contenuto in una cella di memoria e riporre il risultato nella cella successiva. Si suppone che il numero sia minore di 16 e di usare una tabella che contenga i quadrati dei numeri compresi tra 0 e 15, estremi inclusi.

```
INDATO   EQU 200H        ;indirizzo del numero
         ORG 100H
START:
         LD A,(INDATO)
         LD L,A
         LD H,0          ;si costruisce l'indice a 16 bit del
                         ;quadrato cercato
         LD DE,TBQUAD    ;indirizzo base della tabella
         ADD HL,DE       ;indirizzo dell'elemento in tabella
         LD A,(HL)       ;si prende il quadrato del numero
         LD (INDATO+1),A
                         ;e lo si memorizza
         HALT

TBQUAD:  DEFB 0          ;tabella dei quadrati
         DEFB 1
         DEFB 4
         DEFB 9
         DEFB 16
         DEFB 25
         DEFB 36
         DEFB 49
         DEFB 64
         DEFB 81
         DEFB 100
         DEFB 121
         DEFB 144
         DEFB 169
         DEFB 196
         DEFB 255
```